IN-02
43105
p.20

A Fast Sorting Algorithm For A Hypersonic Rarefied Flow Particle Simulation On The Connection Machine

Leonardo Dagum

November 1989

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.44

NASA Cooperative Agreement Number NCC 2-387

Research Institute for Advanced Computer Science
An Institute of the Universities Space Research Association

A Fast Sorting Algorithm For A Hypersonic Rarefield Flow Particle Simulation On The Connection Machine

*Leonard Dagum**

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 89.44
November, 1989

The datas parallel implementation of a particle simulation for hypersonic rarefied flow described by Dagum associates a single parallel data element with each particle in the simulation. The simulated space is divided into discrete regions called "cells" containing a variable and constantly changing number of particles. The implementation requires a global sort of the parallel data elements so as to arrange them in an order that allows immediate access to the information associated with cells in the simulation. This paper describes a very fast algorithm for performing the necessary ranking of the parallel data elements and compares the performance of the new algorithm with that of the microcoded instruction for ranking on the Connection Machine.

Keywords: Connection Machine, Monte Carlo, particle simulation, sorting.

Work reported herein was supported by DARPA via Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

*Department of Aeronautics and Astronautics, Stanford University, Stanford, California. Work performed at Stanford University as as a summer visitor at RIACS.

Introduction

Of increasing interest to NASA and the fluid mechanics community in general is the development of accurate and efficient methods to treat hypersonic rarefied flow problems. Hypersonic flows are typically characterised by a freestream Mach number greater than 4 where the Mach number defined as the ratio of fluid speed to local speed of sound. Rarefied flows are characterised by a large Knudsen number, usually greater than 0.01, where the Knudsen number is defined as the ratio of the local mean free path for the molecules in the fluid to a scale dimension in the flow.

Hypersonic rarefied flow conditions are encountered by flight vehicles operating in the upper atmosphere (altitude 50–150 km) and are of consequence in the design of future vehicles such as the National Aerospace Plane (NASP) and Aero-Assisted Space Transfer Vehicles (ASTV's). The standard method for solving hypersonic rarefied flow problems is through direct particle simulation methods[1−5], however the huge computational capacity required to solve even a modest sized problem of practical interest has severely restricted their use.

Dagum[6] describes the data parallel implementation of a very efficient particle simulation algorithm developed at Stanford University[7−10]. The implementation associates one data processor with each individual particle in the simulation, thus the collision and movement of particles is perfectly load balanced. However, there is a bottleneck associated with identifying partners for collision. This step requires identifying all the particles within each cell in a grid, and this is carried out by sorting particles by order of the cell they occupy and placing all the particles occupying a given cell into a sequence of processors with contiguous addresses. In Dagum's implementation this was accomplished by using the Paris rank instruction[11] which is a microcoded version of the data parallel radix sort described by Hillis and Steele[12]. The present paper describes a faster (three times faster) algorithm for performing this ranking.

Problem Statement

The physical model being simulated consists of a set of particles representative of a gas moving through a representation of a wind tunnel. Computationally, each particle is described by an individual data processor which stores information on the physical state of the particle (i.e. its position and velocity) and the wind tunnel is represented by a grid of cells through which the particles move and from which macroscopic thermodynamic quantities (such as pressure or temperature) can be sampled.

All particles occupying a given cell are candidates for collision and pairs are made of these. In order to make this pairing as well as to sample macroscopic quantities it is expedient to have all the particles in a cell be represented by a contiguous set of data processors. On the Connection Machine this means having those virtual processors which represent the particles in a cell occupy a contiguous set of NEWS addresses[11]. The Connection Machine's firmware allows one to map the hypercube addresses of the processors into a grid topology (called a NEWS grid) of up to 31 dimensions with neighbours on the grid being physical neighbours in the Connection Machine. This arrangement allows very fast communication to occur between neighbouring processors on the grid. The NEWS grid used in this problem is one dimensional, therefore given the starting and ending address of a cell one can identify all the particles within it (see figure 1). Furthermore, the pairing of collision candidates can proceed as "even with odd", that is all the even–addressed processors look to their odd neighbour for collision.

The problem then is one of getting the particles into this convenient order. Clearly if one starts with the particles in order, after one time step particles will have moved from one cell to another and the order is lost. To regain the order one has to have all the processors identify the cell that their particle occupies and then sort the particles by order of this cell position. For this purpose it is necessary to map a two or three dimensional grid of cells to one dimension. The cell's index when mapped to one dimension, which will

be referred to as the "cell index", gets used as the key in the sort.

In using even/odd pairing of collision candidates consideration must be given to the statistical randomness of the pairs. If there are n particles in cell i, then $n(n-1)/2$ different pairings can be made. Using even/odd pairing creates $n/2$ different pairs out of the $n(n-1)/2$ possible. It is important that the $n/2$ pairs made at one time step be statistically independent of the $n/2$ pairs made in the previous time step. One way of ensuring this statistical independence is to concatenate a fixed number of random bits to the end of the cell index and sort on this expanded key[6]. In this way the particles no longer maintain the same relative ordering within a cell and even/odd pairing produces a sample statistically independent from the previous one.

Two Fundamental Observations

In using a generalized sort to solve this problem one is disregarding an abundance of information available for designing a more specific and therefore more efficient sorting algorithm. To this end it is useful to make the following observations regarding the dynamics of the simulation.

(1) On every time step the set begins and ends in an ordered state. The disordering of the particles occurs through their motion from one cell to another. Furthermore, the nature of this motion is such that on one time step only about a third of the particles will change cells, therefore the set is never greatly out of order. In fact it is precisely for this reason that there is statistical dependence between even/odd pairings in succeeding time steps unless an effort is made to enhance the disorder.

(2) The motion of the particles is such that to a very high probability if a particle moves out of its current cell it will move only into its immediately neighbouring cell, that

is, particles do not move more than one cell width per time step (see figure 2).

Using these two observations it is possible to devise a sorting algorithm tailored to this specific problem which is much more efficient than using a generalized sort.

The New Sorting Algorithm

The new sorting algorithm proceeds in the following manner. Making use of the first observation, at the beginning of the time step the set is ordered and every processor is storing a value for its particle's current cell index. The particles then go through their motion after which a new value for the cell index must be computed. Both the old and the new values are stored, and now use is made of the second observation. It is convenient at this point to map the cell index into two dimensions and designate the pre–motion values by i, j and post–motion values by i', j'. Referring to figure 2 and considering the second observation it is obvious that a particle beginning in cell i, j has nine different *and mutually exclusive* possibilities for its new cell location i',j'. (In three dimensions there are 27 mutually exclusive possibilities.) Conversely, if at the end of its motion a particle is occupying cell i',j', there are nine mutually exclusive possibilities for its previous cell position i, j. Therefore one can divide the particles into nine distinct and ordered sets based on the nine distinct possibilities for a previous cell location. In other words, because a particle in cell i',j' has nine mutually exclusive possibilities for its previous cell location i, j, and because the particles were ordered in their previous cells, it follows that the order must be preserved in nine mutually exclusive sets. The problem thus has been reduced to one of identifying these nine ordered sets and merging them into just one set.

Identifying each set is accomplished by simply comparing the previous cell position to the current one. To merge the sets it is necessary to identify in the lowest numbered

processor for every cell in each set, and then enumerate in each set the processors representing a cell (see figure 3). A one dimensional grid, referred to as the "merging grid" and distinct from the physical grid of the simulation, is created with size at least 9 times the number of cells in the simulation. In this way there is a merging grid element for every cell in each set. A send–with–add[11] from the particles to the new grid is used to get the number density for every cell in each set, then a scan–with–add[11] is used to create a running sum of the number density. Each value in the merging grid now is the greatest rank in the merged list for the particles in the cell it handles.

To get the grid result to the processors representing the particles in an efficient manner just one processor in each group representing a cell in each set gets the cell's merged value from the grid, and this value is copied across the rest of the processors in the group in the set. The lowest numbered processor for every cell in each set was identified earlier and is used for this purpose.

Figure 3 is a schematic for the patterns of communication. Steps in the algorithm proceed from left to right across the page. In the first step the nine sets are identified and the particles in each set are enumerated with the enumeration re–starting at every cell. This requires nine distinct pairs of scan operations, a pair for each set. The first scan is necessary to identify cell boundaries in a set and the second scan enumerates the particles in each cell. The next step of the algorithm requires all processors to send to the merging grid to create the cell number density. The running sum is then created using a single scan–with–add. Next, one processor in every cell in every set gets its running sum value from the merging grid. This is depicted in the figure by an arrow with heads on both ends thus emphasizing the fact that this operation requires communication in both directions. Finally, this value is copied across the processors in the cell in each set by using nine distinct scan–with–copy operations. Now the processors can compute their rank simply by subtracting their enumeration within the cell (step 1 of figure 3) from the running sum

of cell number densities in each set.

Maintaining Statistical Independence

It was claimed above that maintaining statistical independence of pairings between time steps is a concern of the simulation. In using the generalized sort it was necessary to concatenate random bits to the end of the key and order the particles on this expanded key. The new algorithm maintains elements of randomization in two ways. The first of these comes about from the manner in which the nine sets are mapped to the merging grid; the second is a result of the manner in which the merging grid running sum is used to compute the rank of a particle.

In mapping the nine sets to the merging grid there are 9! different possibilities. Figure 4 illustrates one such possibility. Consider an arbitrary cell in the simulation and its nine mutually exclusive sources for particles, here numbered 1 through 9. Each of these sources has an element associated with it in the merging grid. The nine elements together account for all the particles in the cell under consideration. It is clear that these nine sources can be mapped to the merging grid through any permutation of 9. By using a random, statistically independent permutation on every time step randomness is introduced to the outcome of the ranking. Once a permutation is chosen it is used in mapping all the cells at that time step, in other words the permutation is a front end array that gets applied in mapping the nine sets of particles to the merging grid.

Unfortunately this does not completely remove the concern with maintaining statistical independence in even/odd pairings between time steps. Empirical measurements show that one can expect on average two thirds of the particles in the simulation to remain in their cells over one time step. Consequently many of the particles in a cell can be

expected to maintain the same neighbours in the NEWS grid between time steps.

Consider a particular cell and let f be the fraction of particles which remain in the cell over the time step. Since the pairing of particles is to proceed as even with odd, it is most desirable in terms of maintaining statistical independence between pairings if the particles which exit the cell were not neighbours in the NEWS grid at the beginning of the time step. If this is true then of the particles which remain in the cell a certain fraction are guaranteed to get paired differently from the previous time step. For example say that only odd numbered particles exit the cell, in the previous time step the pairs were even with odd but in the new time step the pairs created from the particles which did not exit the cell are now "even with succeeding even" since the odd particles in between have left the cell. Therefore in the best case none of the particles which leave the cell are neighbours and of the particles which remain the fraction $\frac{2f-1}{f}$ are neighbours over the time step. In the worst case all the particles which leave the cell are neighbours therefore of the particles which remain all are neighbours over the time step. Typically one can expect a result somewhere between these two extremes.

This might not seem very encouraging, however additional randomization exists in the method. In computing the rank of a particle, its enumeration from the first step of the algorithm is subtracted form the greatest rank of particles in both its set and its cell. This effectively reverses the enumeration of particles, therefore the particles which did not change cells have their order reversed. Consequently, if in a cell an odd number of particles do not move out of the cell, the reversing of their order results in a new and different pairing of these particles. The converse is not true, if the number is even it is still possible for the pairing of these particles to differ between time steps depending on the number of particles ranked below them in the cell.

When Assumptions Fail

The algorithm has been presented from a physical perspective and in the context of a generic cell in a generic time step. Two observations of the dynamics of the simulation were necessary for the algorithm to be valid. It is necessary now to discuss the situations where these observations do not hold true and the algorithm cannot be used.

The first observation claimed that the particles go from an ordered to a disordered state through their motion from one cell to another. This is not true at the upstream boundary of the wind tunnel where new particles must be introduced to maintain the freestream. However the introduction of new particles can be delayed an arbitrary number of time steps[6] therefore it is convenient to employ the generalized sort on those time steps where new particles are introduced and use the fast sort on the other time steps. The generalized sort also thoroughly shuffles the order of the particles and it is reassuring to have such a shuffle occurring periodically throughout the calculation.

Using the generalized sort periodically in this manner also is important if the second observation fails to be true. A crucial assumption for the new sort to be valid is that particles do not move beyond their immediately neighbouring cell in one time step. This is true to a very high probability, however given the statistical nature of the simulation it is impossible to rule out the possibility of a particle not holding to this assumption. The outcome of such a situation is not catastrophic, however there does result an incorrect ordering of the particles. If the order is not restored the first observation becomes untrue and the sort fails on succeeding time steps. Therefore the order continues to deteriorate until it gets restored via a generalized sort.

Results and Discussion

The algorithm described here was implemented on the 32k processor Connection Machine Model 2 at the NASA Ames Research Center. The code for it was written fully in C/Paris and employed in the simulation of hypersonic flow over a wedge. In figure 5 the performance of the new algorithm is compared to that of the Paris rank instruction. Figure 5a is a plot of the computational time just to rank the particles and does not include the time for moving particles into their sorted order. One can see from this figure that the new ranking algorithm is about three times faster than the Paris rank. These times were measured using just 8k physical processors but are fixed by virtual processor ratio so can be scaled accordingly for greater numbers of physical processors.

Dagum[6] gives performance results only for the full sort, i.e. he includes the time to move the particles into their sorted order. Figure 5b compares these times for the old and new algorithms. The new sort takes about 40% of the time of the old sort. Recall, however, that the old sort must still be used periodically, typically once every seven time steps. If this is included into the performance figure then over a full flow solution one can expect the new sorting algorithm to take 50% of the time of the old sorting algorithm.

The algorithm has been presented and discussed for only two dimensions. The extension to three dimensions is straightforward but does involve a loss in performance due to increased communications. In three dimensions there are 27 mutually exclusive sets instead of 9. The algorithm requires three scan operations per set, in two dimensions these operations account for 55% of the time to rank. In the worst case, in three dimensions that fraction of the algorithm would triple in time so overall the new ranking algorithm would take about 2.1 times longer in three dimensions. This would still be better than the Paris rank even if the number of cells is held fixed. Since in three dimensions one would expect to have more cells it is reasonable to assume that the Paris rank would also be slower though not as dramatically so.

Conclusions

By careful consideration of the physics behind the direct simulation of a hypersonic rarefied flow, it has been possible to design a very fast algorithm to perform the ranking of processors necessary in the adaptive domain decomposition of this problem. The new ranking algorithm is three times faster than the PARIS instruction for ranking and brings about a significant improvement in performance for the whole simulation.

Acknowledgements

I gratefully acknowledge Professor Donald Baganoff for his guidance and insight throughout the course of the work.

The work reported here was supported in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Administration (USRA), by NASA under Hypersonic Training and Research grant NAGW-965 and by the Air Force Office of Scientific Research under grant AFOSR 88-0139.

References

[1] Bird, G.A., *Molecular Gas Dynamics*, Oxford University Press, London, 1976.

[2] Bird, G.A., "Monte Carlo Simulation of Gas Flows," *Annual Review of Fluid Mechanics,* Vol. 10, pp.11-31, 1978.

[3] Bird, G.A., "Monte Carlo Simulation in an Engineering Context," *Prog. in Astro. and*

Aero., Vol. 74, pp. 239–255, 1981.

[4] Bird, G.A., "Direct Simulation of Gas Flows at the Molecular Level," *Proceedings of the First World Congress on Computational Mechanics,* The University of Texas at Austin, September 22–26, 1986.

[5] Derzko, N.A., "Review of Monte Carlo Methods in Kinetic Theory," *UTIAS Review,* No. 35, Univ. of Toronto, 1972.

[6] Dagum, L., "Implementation of a Hypersonic Rarefied Flow Particle Simulation on the Connection Machine," *Proceedings-Supercomputing '89, Nov 13-17, 1989, Reno NV.*

[7] McDonald, J.D., Baganoff, D. "Vectorization of a Particle Simulation Method for Hypersonic Rarefied Flow," *AIAA-88-2735* from AIAA Thermophysics, Plasmadynamics and Lasers Conference, San Antonio, June 27–29, 1988.

[8] Feiereisen, W., McDonald, J.D., "Three Dimensional Discrete Particle Simulation of an AOTV," *AIAA-89-1711* from AIAA 24^{th} Thermophysics Conference, Buffalo, June 12–14, 1989.

[9] Woronowicz, M.S., McDonald, J.D., "Application of a Vectorized Particle Simulation in High-Speed Near-Continuum Flow," *AIAA-89-1665* from AIAA 24^{th} Thermophysics Conference, Buffalo, June 12–14, 1989.

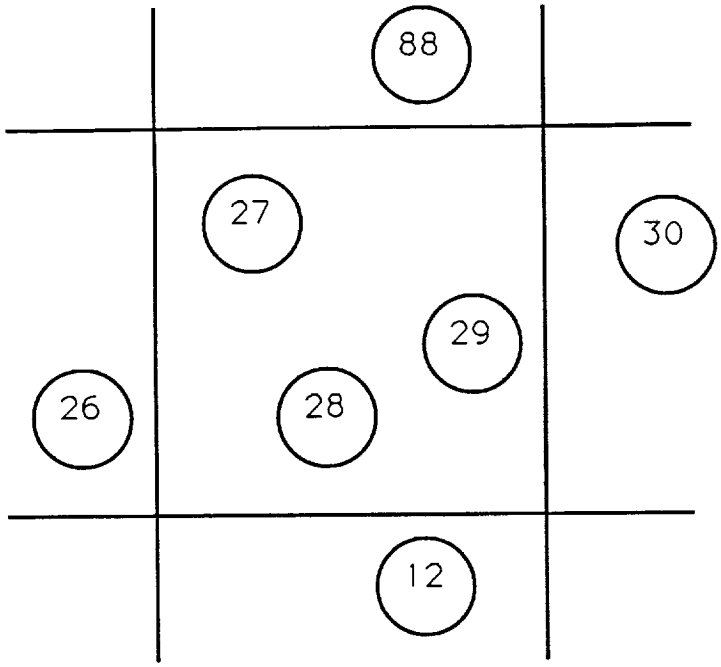
[10] Baganoff, D., McDonald, J.D., "A Collision-Selection Rule for a Particle Simulation Method Suited to Vector Compilers," submitted to *Physics of Fluids*, August 1989.

[11] Thinking Machines Corp., *The Connection Machine System-Paris Reference Manual Version 5.0A Field Test*, June 1988.

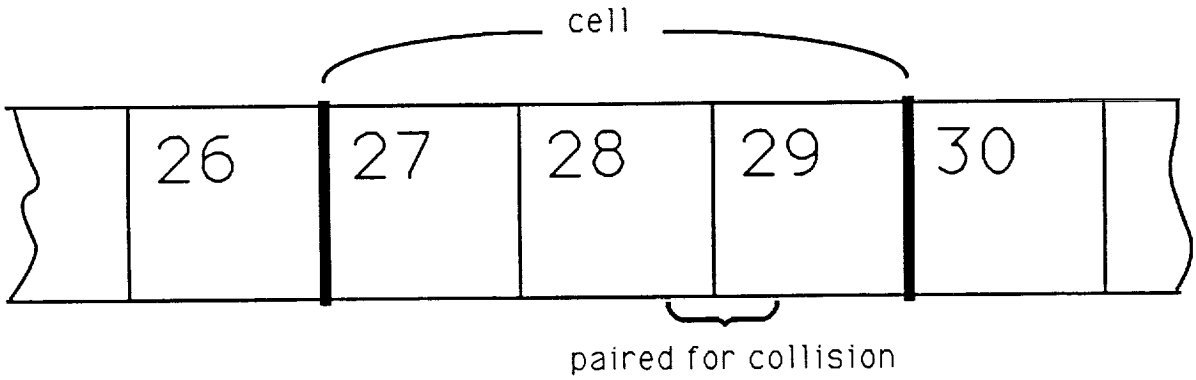
[12] Hillis, W.D., Steele, G.L., "Data Parallel Algorithms," *Communications of the ACM,* Vol. 29, No.12, pp. 1170–1183, 1986.

[16] Knuth, D.E., *The Art of Computer Programming*, Vol. 2, 2^{nd} ed., pp. 139–140, Addison-Wesley, Reading, MA, 1973.

[17] Aldous, D., Diaconis, P., "Shuffling Cards and Stopping Times," *American Mathematical Monthly,* Vol. 93, No. 5, pp. 333–348, 1986.



Physical Space



Computational Space

Figure 1. Physical and computational space for the particle simulation. Particles occupying the same cell in physical space are represented by neighbouring processors in computational space.

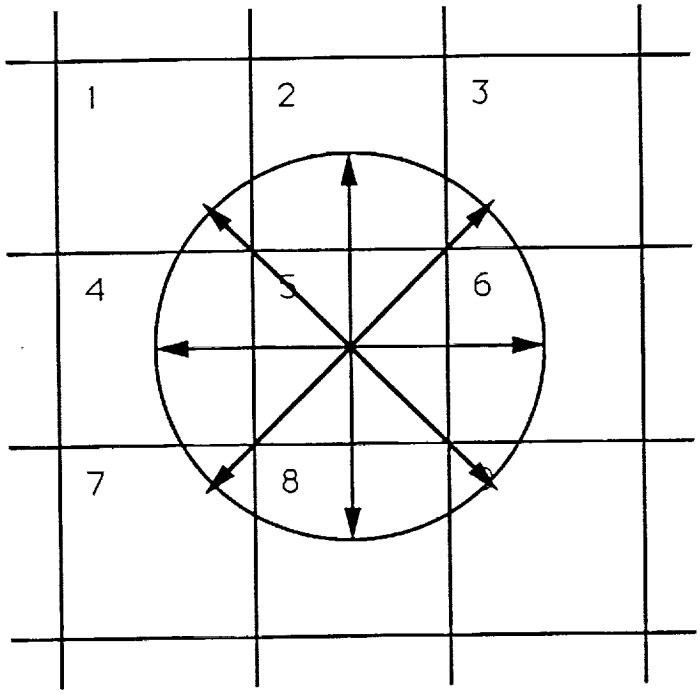


Figure 2. The maximum radius of motion over one time step is to a very high probability less than one cell width. Particles move only into their immediately neighbouring cells.

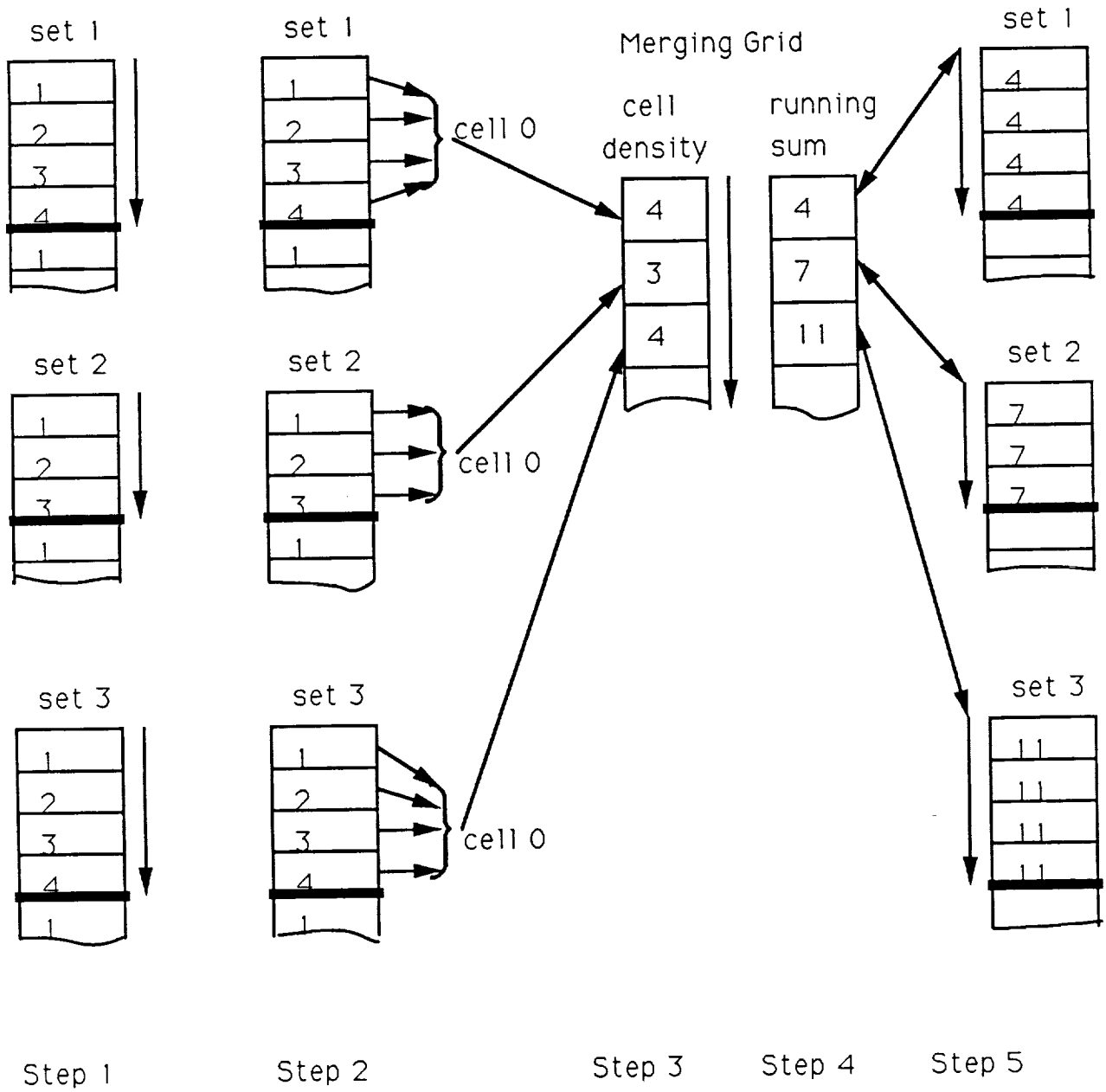


Figure 3. Schematic of the communication patterns in the new sorting algorithm.

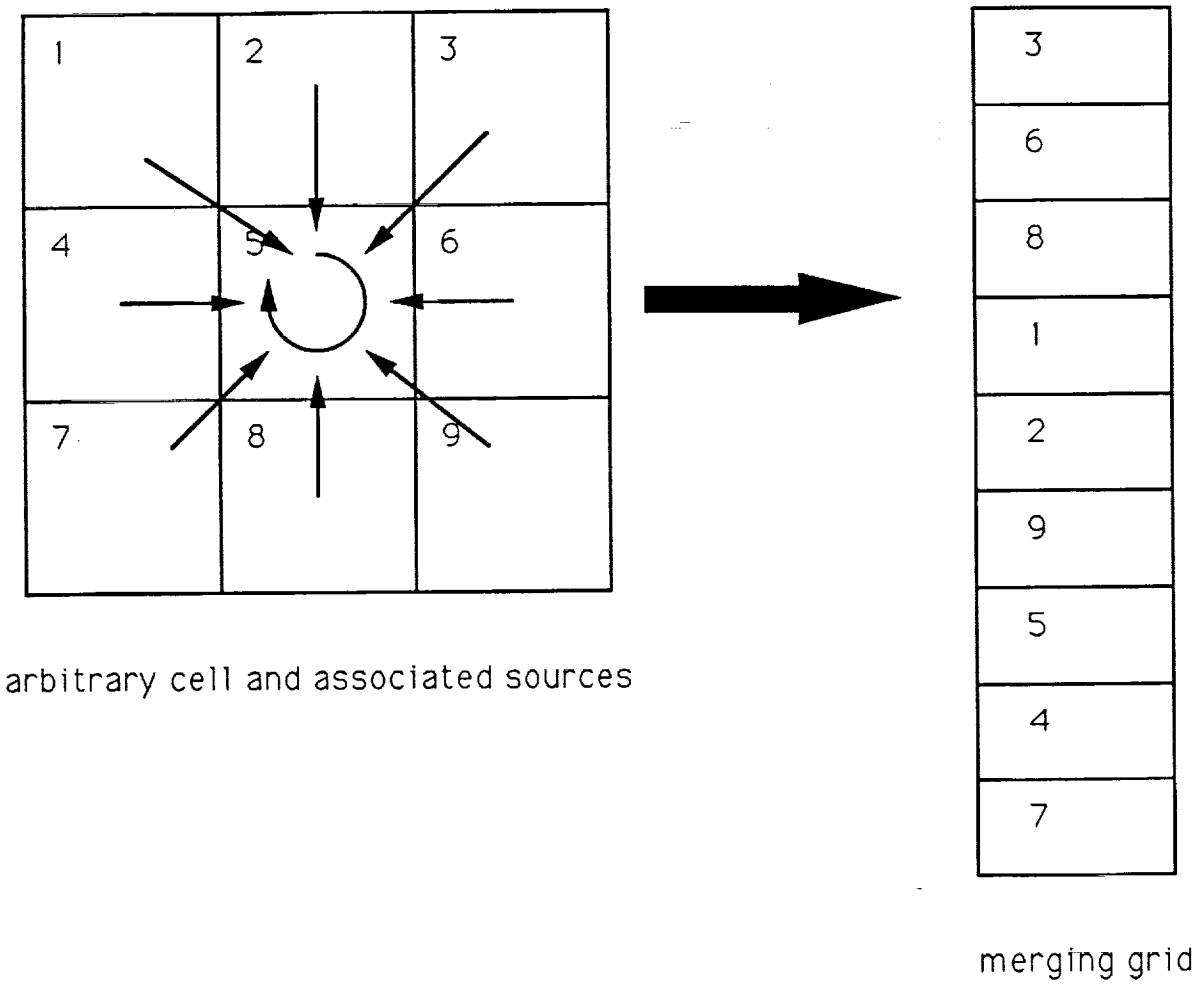


Figure 4. One possible mapping of the nine sources to the merging grid. Each cell has nine sources for incoming particles which taken for all the cells make up nine mutually exclusive ordered sets of particles. There are 9! different possibilities for mapping these sources to the merging grid.

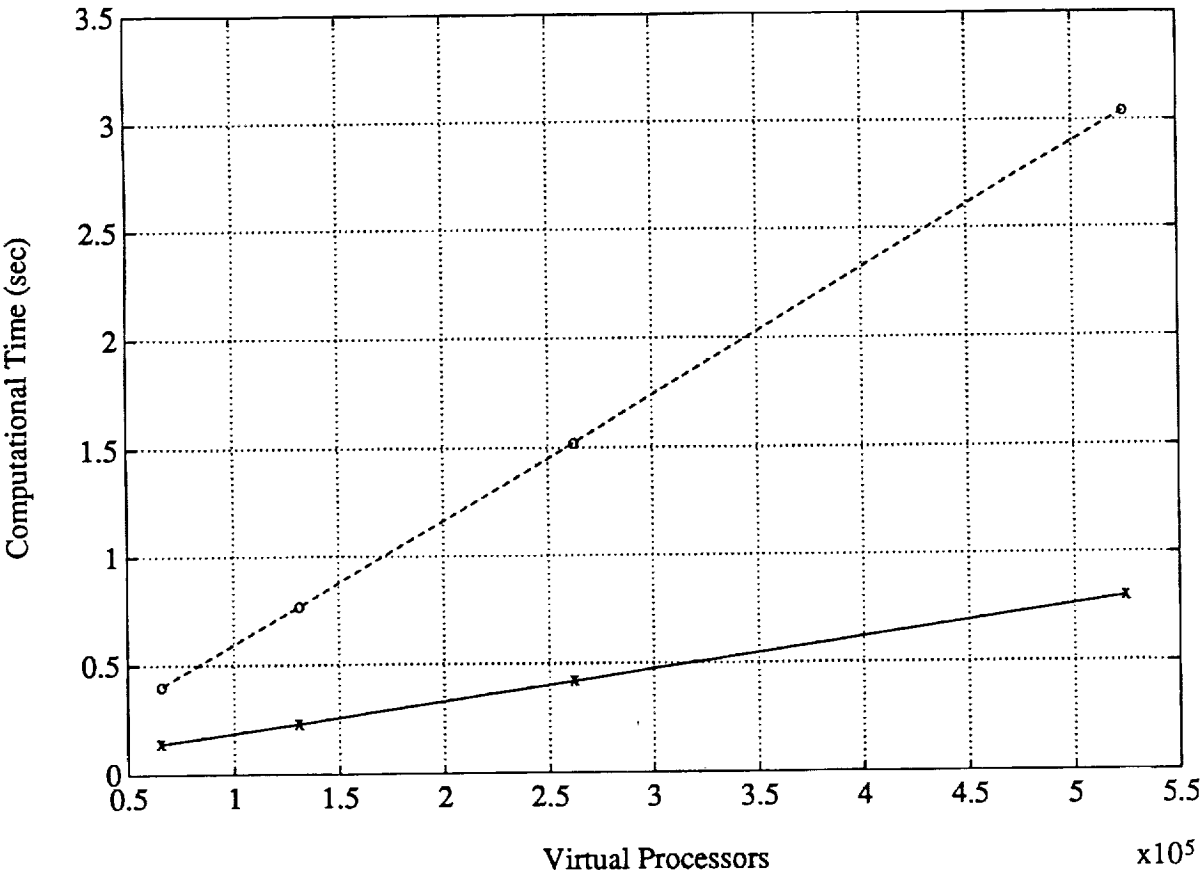


Figure 5a. Time to rank with Paris rank (o) and new algorithm (x).

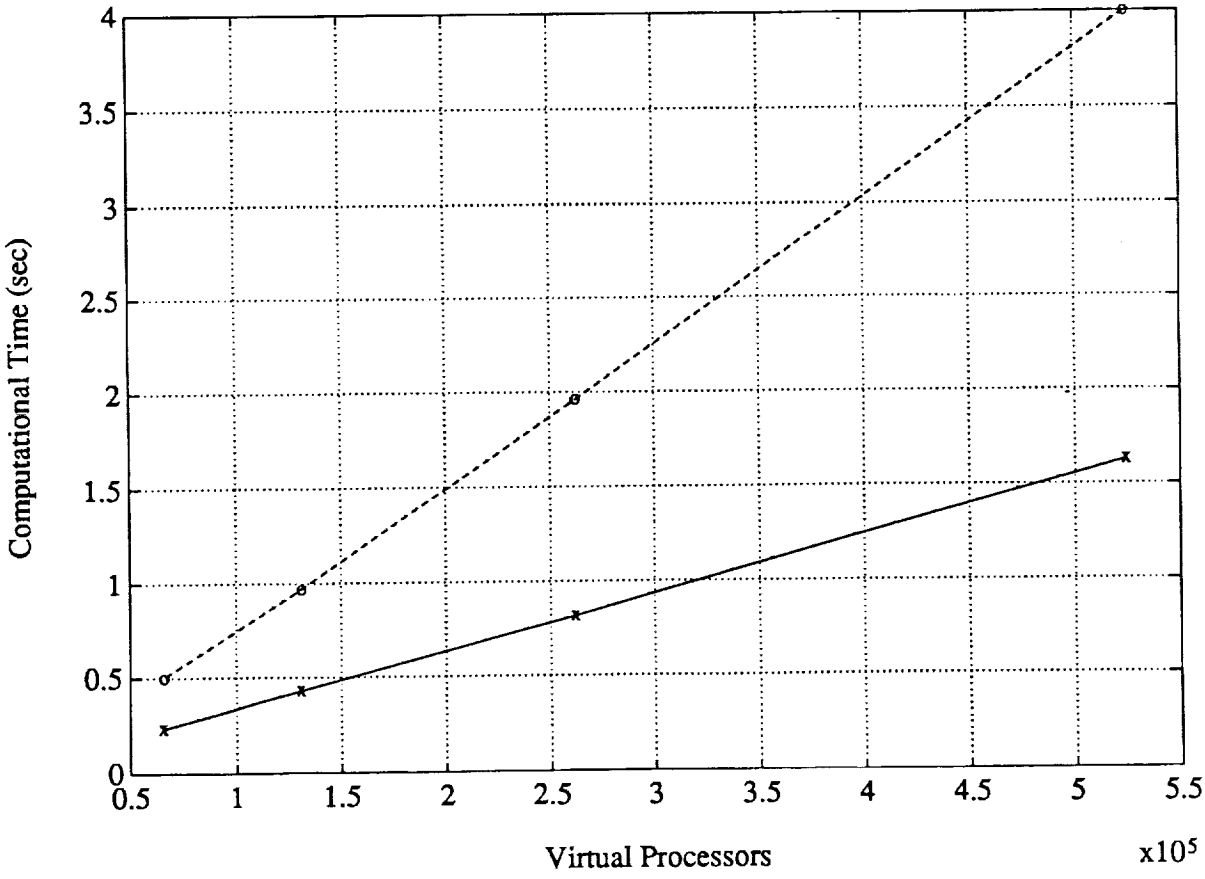


Figure 5b. Time to sort with Paris rank (o) and new algorithm (x).